science.sciencemag.org/cgi/content/full/science.aaw4741/DC1

Supplementary Material for

Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations

Maziar Raissi, Alireza Yazdani, George Em Karniadakis*

*Corresponding author. Email: george_karniadakis@brown.edu

Published 30 January 2020 as *Science* First Release
DOI: 10.1126/science.aaw4741

This PDF file includes:

Materials and Methods
Supplementary Text
Figs. S1 to S21
Tables S1 to S4
References

Other Supplementary Material for this manuscript includes the following:
(available at science.sciencemag.org/content/science.aaw4741/DC1)

Movies S1 and S2

Materials and Methods

Starting with the transport equation, we can rewrite it in the following non-dimensional form

$$c_t + uc_x + vc_y + wc_z = \mathrm{Pe}^{-1}(c_{xx} + c_{yy} + c_{zz}), \tag{S1}$$

which governs the evolution of the normalized concentration $c(t, x, y, z)$ of a passive scalar transported by an incompressible Newtonian fluid whose dynamics are described by the Navier-Stokes and continuity equations (also non-dimensional) given below

$$\begin{array}{l} u_t + uu_x + vu_y + wu_z = -p_x + \mathrm{Re}^{-1}(u_{xx} + u_{yy} + u_{zz}), \\ v_t + uv_x + vv_y + wv_z = -p_y + \mathrm{Re}^{-1}(v_{xx} + v_{yy} + v_{zz}), \\ w_t + uw_x + vw_y + ww_z = -p_z + \mathrm{Re}^{-1}(w_{xx} + w_{yy} + w_{zz}), \\ u_x + v_y + w_z = 0. \end{array} \tag{S2}$$

Here, Re and Pe are the Reynolds and Péclet numbers, defined, respectively as UL/ν and UL/κ, where L, U are the characteristic length and velocity, and ν, κ are the kinematic viscosity and mass diffusion coefficient, respectively. One question that would naturally arise is whether the information on the passive scalar in the training domain and near its boundaries is sufficient to result in a unique velocity field. The answer is that normally there are no guarantees for unique solutions unless proper boundary conditions are explicitly imposed on the domain boundaries. However, as shown later for the benchmark problems studied in the current work, an informed selection of the training boundaries in the regions where there are sufficient gradients in the concentration of the passive scalar could possibly eliminate the requirement of imposing velocity and pressure boundary conditions.

By extending our earlier work on physics-informed deep learning (*5*) and *deep hidden physics models* (*20*), we approximate the function

$$(t, x, y, z) \longmapsto (c, u, v, w, p)$$

by a physics-uninformed deep neural network and construct the following *Navier-Stokes in-*

formed neural networks (see Fig. S1) corresponding to Eqs. S1 and S2 i.e.,

$$\begin{aligned}
e_1 &:= c_t + uc_x + vc_y + wc_z - \text{Pe}^{-1}(c_{xx} + c_{yy} + c_{zz}),\\
e_2 &:= u_t + uu_x + vu_y + wu_z + p_x - \text{Re}^{-1}(u_{xx} + u_{yy} + u_{zz}),\\
e_3 &:= v_t + uv_x + vv_y + wv_z + p_y - \text{Re}^{-1}(v_{xx} + v_{yy} + v_{zz}),\\
e_4 &:= w_t + uw_x + vw_y + ww_z + p_z - \text{Re}^{-1}(w_{xx} + w_{yy} + w_{zz}),\\
e_5 &:= u_x + v_y + w_z.
\end{aligned} \tag{S3}$$

We compute the required derivatives to construct the residual networks e_1, e_2, e_3, e_4, and e_5 by applying the chain rule for differentiating compositions of functions using automatic differentiation (*21*). A schematic representation of the resulting *Navier-Stokes informed neural networks* is given in Fig. S1. It is worth emphasizing that automatic differentiation[1] is different from, and in several respects superior to, numerical or symbolic differentiation – two commonly encountered techniques of computing derivatives. In its most basic description (*21*), automatic differentiation relies on the fact that all numerical computations are ultimately compositions of a finite set of elementary operations for which derivatives are known. Combining the derivatives of the constituent operations through the chain rule gives the derivative of the overall composition. This allows accurate evaluation of derivatives at machine precision with ideal asymptotic efficiency and only a small constant factor of overhead. In particular, to compute the required derivatives we rely on Tensorflow (*22*), which is a popular and relatively well documented open-source software library for automatic differentiation and deep learning computations. In TensorFlow, before a model is run, its computational graph is defined statically

[1]We emphasized "automatic differentiation" several times in the paper. Here, we briefly explain what "automatic differentiation" is, and particularly, explain how one would calculate the derivative du/dx, where x is an input to the neural network and u is one of its outputs. Since the network is constructed as a sequence of simple and analytically differentiable functions (linear combinations and activation functions), it is possible to apply the chain rule backward (starting at u and going back through each layer of the network until x is reached) or forward (starting at x and going forward through each layer of the network until u is reached) to find a numerical value for the partial derivative of u with respect to x. The key technical subtlety here is to "memoize" some of the operations while going forward and backward through each layer of the network. Memoization is a common optimization technique in computing used primarily to speed up computer programs by storing the results of expensive function calls and returning the cached result when the same inputs occur again. While automatic differentiation is now readily available in most scientific computing platforms (e.g., TensorFlow, PyTorch, Matlab, etc.), it is still a very good exercise to code one up from scratch (see e.g., `https://github.com/maziarraissi/backprop`) to gain more intuition.

rather than dynamically as for instance in PyTorch (*23*). This is an important feature as it allows us to create and compile the computational graph for the Navier-Stokes informed neural networks (Eq. **S3**) only once and keep it fixed throughout the training procedure. This leads to significant reduction in the computational cost of the proposed framework.

The shared parameters of the *physics-uninformed* neural networks for c, u, v, w, p and the *physics-informed* ones e_1, e_2, e_3, e_4, e_5 can be learned by minimizing the mean squared error loss function defined in Eq. **1**. The first term in the equation corresponds to the training data $\{t^n, x^n, y^n, z^n, c^n\}_{n=1}^N$ on the concentration $c(t, x, y, z)$ of the passive scalar, while the last term penalizes the structure imposed by Eqs. **S1** and **S2** at a finite set of *residual points* $\{t^m, x^m, y^m, z^m\}_{m=1}^M$ whose number and locations can be different from the actual training data.

To generate high-resolution datasets for different benchmark problems investigated in the current work, we have employed the spectral/hp-element solver Nektar/Nektar++ in which the Navier-Stokes Eqs. **S2** along with the transport Eq. **S1** are approximated using high-order semi-orthogonal Jacobi polynomial expansions (*7*). The numerical time integration is performed using a third-order stiffly stable scheme until the system reaches its stationary state. In what follows, a small portion of the resulting dataset corresponding to this stationary solution will be used for model training, while the remaining data will be used to validate our predictions. Our algorithm is agnostic to the choice of initial and boundary conditions as well as the geometry. However, we provide detailed information for every benchmark problem for the sake of completeness and reproducibility of the numerically generated data[2]. In particular, concentration data are produced by forward numerical simulations for the benchmark problems investigated

[2]Nektar++ is the spectral/hp element and open-source solver that can be downloaded from `https://www.nektar.info/downloads`. The source code for Nektar along with the grids and input files for generating the data are available at `https://github.com/alirezayazdani1/HFM`. Furthermore, the input data and the codes for ML training and inference used in this manuscript are publicly available on GitHub at `https://github.com/maziarraissi/HFM`.

in this work using the open-source spectral/hp element solvers (Nektar/Nektar++), where the degrees of freedom (DoF), timestep size, number of CPU processors, and the number of timesteps to reach the final solution are reported in Table **S1**. Note that a higher DoF i.e., the total number of spectral modes in Table **S1**, is required to resolve the flow dynamics accurately, which leads to longer simulation times. Moreover, the timestep size is typically much smaller than the temporal resolution of the concentration data that we use to train the HFM. Normally, forward simulations are performed from a zero-initial condition until the final solution is reached after a number of timesteps. This is not the case for the HFM and inference as any time window can be used for data acquisition and predictions.

To obtain the results reported in this manuscript, we represent each of the output functions c, u, v, w, and p by a 10-layer deep neural network with 50 neurons per hidden layer (see Fig. **S1**). To argue for making the networks deeper rather than wider, one could draw analogies from the circuit theory where there are, in fact, mathematical proofs showing that for some functions very shallow circuits require exponentially more circuit elements to compute than the deep circuits do (*24*). Moreover, the computational complexity of a neural network grows linearly with the depth of the network. As for the activation functions, we use the swish (*25*) activation function given explicitly by

$$\text{swish}(x) := x \ \text{sigmoid}(x) = x/(1 - \exp(-x)).$$

Currently, the most successful and widely-used activation function for deep learning is the Rectified Linear Unit (ReLU) given by $\max(0, x)$. However, when it comes to physics-informed neural networks (see Eq. **S3**), we cannot use this particular activation function because it leads to wiggles in the computed first order derivatives with respect to the input variables (e.g., t, x, y, z) and causes the second-order derivatives with respect to the inputs to vanish altogether (see Fig. **S2**). Therefore, we use the swish activation function as a smoothed version of ReLU.

In TensorFlow, the current implementations of activation functions such as $\tanh$ and $\sin$ are slightly faster than the swish activation we are adopting here [3]. In general, the choice of a neural network's architecture (e.g., number of layers/neurons and form of activation functions) is crucial and in many cases still remains an art that relies on one's ability to balance the trade-off between *expressivity* and *trainability* of the neural network (*26*). Our empirical findings so far indicate that deeper and wider networks are usually more expressive (i.e., they can capture a larger class of functions) but are often more costly to train (i.e., a feed-forward evaluation of the neural network takes more time and the optimizer requires more iterations to converge). However, these observations should be interpreted as conjectures rather than as firm results[4]. In this work, we have tried to choose the neural networks' architectures in a consistent fashion throughout the manuscript. However, there might exist other architectures that could possibly improve some of the results reported in the current work. Hyper-parameter tuning techniques, and more broadly meta-learning can be used to automatically tune the network and even search for more efficient architectures in future studies. Furthermore, to accelerate the training process, we employ weight normalization (*27*): a reparameterization of the weight vectors in a neural network that decouples the length of the weight vectors from their direction. Such a reparameterization is inspired by batch normalization (*28*) but does not introduce any dependencies between the examples in a minibatch. Batch normalization is another widely used and successful technique in deep learning, which we are unable to use here as it interferes with the physics of the problem making the equations batch dependent. For this reason, we use weight normalization as an alternative to batch normalization to accelerate the training of physics-informed neural networks.

[3]It is worth mentioning that our algorithm is robust to the choice of the activation function. Please refer to the preprint version of this manuscript on arXiv (`https://arxiv.org/abs/1808.04327`), where we used the $\sin$ activation function.

[4]We encourage the interested reader to check out the codes corresponding to this paper on GitHub at `https://github.com/maziarraissi/HFM` and experiment with different choices of the neural networks' architectures.

As for the training procedure (see e.g., Fig. S7), the results reported in the current manuscript are consistently obtained after 10^6 iterations of training for both two and three dimensional problems using the Adam optimizer (*29*) with *default* hyper-parameters and a learning rate of 10^{-3}. However, our algorithm as reported in the following can converge to solutions with acceptable accuracy in 10^5 iterations of training for both two and three dimensional problems. The mini-batch of data, as well as the residual points used to penalize the equations, processed per each iteration of the Adam optimizer has a size of $10,000$. Every 10 iterations of the optimizer takes around 1.5 and 2.5 seconds, respectively, for two and three dimensional problems on a single NVIDIA Titan V GPU. It is desirable to report the training periods in terms of a metric that is independent of the hardware; the commonly accepted metric in the deep learning literature is "epochs". An epoch corresponds to one pass through the entire dataset. However, we are using an unconventional deep learning platform in the sense that we work with two different datasets rather than one, namely the observation data on the concentration and the residual points used to penalize the equations. It is not clear which one of the two we should use to report the duration of training in terms of epochs. Consequently, in the current manuscript, we adopt the number of iterations as a metric that is independent of the hardware. Moreover, the number of data points for the concentration of the passive scalar for each benchmark problem is stated explicitly below. In this work, we have consistently used a fixed training procedure for all benchmark problems. There might exist other training procedures that could potentially improve the results reported in the current manuscript. For instance, although not pursued here, our experience so far suggests that during training physics-informed deep neural networks, it is often useful to reduce the learning rate as training progresses.

To measure the accuracy of the predictions of our algorithm, we report the relative L_2 error between the regressed function f and the reference function g defined as

$$\mathcal{E}(f,g) := \left(\frac{1}{N}\sum_{i=1}^{N}[f(x_i)-g(x_i)]^2\right) / \left(\frac{1}{N}\sum_{i=1}^{N}\left[g(x_i)-\frac{1}{N}\sum_{i=1}^{N}g(x_i)\right]^2\right), \qquad \textbf{(S4)}$$

where $\{x_i : i = 1, \dots, N\}$ are points scattered in the domain of interest. The above definition has the favorable properties that it is invariant under shift and scaling of both the regressed and the reference functions; i.e., $\mathcal{E}(f+\alpha, g+\alpha) = \mathcal{E}(f,g)$ and $\mathcal{E}(\beta f, \beta g) = \mathcal{E}(f,g)$, for any constants α and $\beta \neq 0$.

In order to verify the robustness of our algorithm to measurement noise, we adopt the commonly accepted approach of "adding artificial white noise" (see e.g., (*30*)) and distort the innate variations (i.e., the variance) of the input data on the concentration of the passive scalar. Specifically, we add white noise with magnitude equal to a given percentage of the standard deviation of the concentration data. Noting that shifting and scaling the concentration will not change the transport Eq. **S1** for c, the algorithm is essentially learning from the variations in the input data. Therefore, it makes sense to add the white noise at level a to the concentration data according to $c + a\ \text{Std}[c]\ \epsilon$, where ϵ is an independent Gaussian noise with zero mean and unit variance i.e., $\text{Var}[\epsilon] = 1$. The following simple argument will then show that we are, in fact, distorting the innate variations of the concentration profile c by $(1 + a^2)$;

$$\text{Var}[c + a\ \text{Std}[c]\ \epsilon] = \text{Var}[c] + a^2\ \text{Var}[c]\ \text{Var}[\epsilon] = \text{Var}[c](1 + a^2), \qquad \textbf{(S5)}$$

regardless of the magnitude (i.e., mean value) of c since $\mathbb{E}[c + a\ \text{Std}[c]\ \epsilon] = \mathbb{E}[c]$.

Supplementary Text

External 2D flow past a circular cylinder

As the first example, we consider the prototypical problem of a two dimensional flow past a circular cylinder, known to exhibit rich dynamic behavior and transitions for different regimes of the Reynolds number $\text{Re} = UD/\nu$. Assuming a non-dimensional free stream velocity $U = 1$, cylinder diameter $D = 1$, and kinematic viscosity $\nu = 0.01$, the system exhibits a periodic steady-state behavior characterized by an asymmetrical vortex shedding pattern in the wake of the cylinder, known as the von Kármán vortex street (*4*). Importantly, the passive scalar is injected at the inlet within the interval [-2.5, 2.5] using a step function (see Fig. S3). The choice of this boundary condition only depends on the region of interest in which velocity and pressure fields are inferred using the concentration field. Furthermore, a constant value of diffusivity for the passive scalar is assumed to be given by $\kappa = 0.01$ resulting in $\text{Pe} = UD/\kappa = 100$. To generate synthetic data using numerical solvers, we use zero-slip and zero-concentration boundary conditions on the cylinder wall. Whereas the Péclet number is chosen to be equal to the Reynolds number, there is no restriction on its value as shown for the internal flow benchmark problem below. Physically, for the flow of gases such as air with smoke as a passive scalar $\text{Pe} \approx \text{Re}$. This is not, however, the case for liquid flows with dye as the passive scalar since diffusivities of dyes are typically smaller than most fluids, which leads to higher Péclet numbers.

A representative snapshot of the input data on the concentration field is shown in Fig. S3. In terms of training data, the input to the algorithm is a point cloud of data on the concentration of the passive scalar $\{t^n, x^n, y^n, z^n, c^n\}_{n=1}^{N}$, scattered in space and time (see e.g., Fig. 2B). As illustrated in this example, the shape and extent of the boundaries of the training domains that we choose for our analysis could be *arbitrary* and may vary by problem. Here, no input information other than the concentration field for the passive scalar c is passed to the algorithm.

We present the relative L_2 errors for the flow problem with arbitrary domain in Fig. S4, where the errors for each field are computed within the prediction time window of $t = 132.08 - 148.08$. Furthermore, we performed an extensive systematic study with respect to the spatio-temporal resolution of the training data and we report the results in Fig. S5. The proposed algorithm tends to be very robust with respect to the spatio-temporal resolution of the point cloud of data. This result is very encouraging when it comes to future applications of the algorithm to realistic visualization data that may suffer from noise and/or low resolutions. For this particular benchmark problem, the errors tend to increase at around 13 snapshots for the resolution in time, which is a very coarse resolution considering that the whole time window is about 2.5 vortex shedding cycles.

To investigate the effect of noise on the model inference, we corrupt the concentration data with additive Gaussian noise according to Eq. S5. Fig. S6 presents the results on the relative L_2 error between the model output and the reference fields for the arbitrary training domain as a function of the level of the noise denoted by parameter a (see Eq. S5). We observe excellent robustness in the algorithm to strong noise levels as high as 160%, which cause at most three-fold increase in the relative L_2 errors. We believe that enforcing the equations at arbitrarily many residual points has a strong regularization effect that makes the algorithm robust.

Furthermore, as depicted in Fig. S7, the sum of mean squared errors given in Eq. 1 seems to be very effective in avoiding the trivial local optima such as the zero and constant solutions to the Navier-Stokes equations. This loss function essentially acts as an *upper bound* for individual losses of equations and data, and eventually leads to optimized values for the individual losses, given enough training time and given that neural networks are universal function approximators (see e.g., (*31*)). However, developing better loss functions or proper weighting of the individual losses to accelerate the training process seems to be necessary in future research. Here, ideas from the emerging fields of transfer learning and multitask learning could be explored (*32*).

Next, we choose the region of interest to contain the cylinder (see the rectangular domain in Fig. S3) so that the fluid forces acting on the cylinder can be inferred. As shown in figure S8, the algorithm is capable of accurately reconstructing the velocity and the pressure fields without having access to sufficient observations of these fields themselves. Note that we use a very small training domain that cannot be used in classical computational fluid dynamics to obtain accurate solutions of the Navier-Stokes equations. Further, other than the velocity on the left boundary, no other boundary conditions are given to the algorithm. We need to impose a Dirichlet boundary condition for the velocity at the left boundary simply because the observations of the concentration are not providing sufficient information (i.e., gradients) in front of the cylinder. More notably, there is no need to impose the no-slip boundary condition on the cylinder wall. The presence of the normal concentration gradient naturally allows our algorithm to infer the zero velocity condition on the walls. In regards to the predicted pressure field, we note that due to the nature of the Navier-Stokes equations for incompressible flows, the pressure field is only identifiable up to an additive constant since only the gradients of the pressure are present in the Navier-Stokes equations.

The regressed pressure and velocity fields can be used to obtain the lift and drag forces exerted on the cylinder by the fluid as shown in Fig. S9. Additionally, one could provide the algorithm with information such as data on the velocities or pressure (e.g., no-slip velocity boundary conditions on the cylinder wall). Consequently, in Fig. S9, we explored the effect of including the no-slip boundary condition, which led to more accurate estimates for the lift and drag forces. The fluid forces on the cylinder are functions of the pressure and velocity gradients. Having trained the neural networks, we can use

$$F_L = \oint \left[-pn_y + 2\text{Re}^{-1}v_y n_y + \text{Re}^{-1}\left(u_y + v_x\right)n_x\right] ds,$$

$$F_D = \oint \left[-pn_x + 2\text{Re}^{-1}u_x n_x + \text{Re}^{-1}\left(u_y + v_x\right)n_y\right] ds,$$

to obtain the lift and drag forces, respectively. Here, $\boldsymbol{n} = (n_x, n_y)$ is the outward normal on the cylinder wall and ds is the differential area on the surface of the cylinder. We use the trapezoidal rule to estimate these integrals. Note that the lift and drag coefficients are defined by $C_L = F_L/0.5\rho U^2$ and $C_D = F_D/0.5\rho U^2$, respectively, which are plotted in Fig. **S9**, where $\rho = 1$ and $U = 1$. Interestingly, the predicted lift and drag coefficients are in good agreement with the reference, both in terms of the frequency of oscillations and the amplitude. Some discrepancy, however, can be observed for the few initial and final time instants, which can be attributed to the lack of data. This can be further clarified by the relative L_2 errors shown in Fig. **S10**. Lack of training data on c for $t < 132.08$ and $t > 148.08$ leads to weaker neural network predictions near the beginning and end points of the training time window. Thus, one should take this into consideration when inference is required within a certain time interval. Note that to compute lift and drag forces, the gradient of the velocity has to be computed on the wall, which can be done analytically using the parametrized surrogate velocity field (i.e., the neural networks). Although no numerical differentiation is needed to compute the gradients, the integration of forces on the surface of the cylinder is approximated by a numerical quadrature. In Fig. **S10**, we also study the effect of including the no-slip boundary condition on the cylinder wall, which leads to more accurate regressions of the fluid velocity and pressure fields but a less accurate regression of the concentration field.

In addition to the velocity and pressure fields, it is possible to discover other unknown parameters of the flow field such as the Reynolds and Péclet numbers. Although these parameters were prescribed in the 2D flow past the cylinder example, we have tested a case in which both parameters are free to be learned by the algorithm. The results are given in Table **S2**, which shows very good agreement with the exact values. From the practical standpoint, the passive scalar diffusivity and the fluid viscosity (hence, their ratio the Prandtl number $\mathrm{Pr} \equiv \nu/\kappa$) may be known in advance. Therefore, discovering the Reynolds number would be sufficient whereas the Péclet number can be computed by Pe = Re Pr.

We note here that the synthetic training data for the passive scalar are generated using DNS with the assumption of zero-Dirichlet $c = 0$ boundary condition on the cylinder wall. A more realistic boundary condition, however, for c would be zero-Neumann $\partial c/\partial n = 0$ on the solid wall. At low Péclet number and uniform concentration profile at the inlet, the zero-Neumann condition will not produce sufficient mixing in the vicinity of the cylinder and hence HFM may not work accurately or could even break down for this case. However, in most realistic settings of experimental fluid mechanics, streaks of passive scalar (e.g., smoke or dye) are injected to the flow free stream at specific points, which are visualized downstream once they reach the bluff body (see e.g., Fig. 72 in (*33*) for an example of streaklines around an airfoil). We have investigated the performance of HFM for both types of boundary conditions with streaks of passive scalar injected into the flow stream at the inlet. We are particularly interested in the HFM predictions for the lift and drag coefficients given a sequence of snapshots of the streaklines around the circular cylinder (see Fig. **S11**A,B)[5]. The results are shown in Fig. **S11**C,D for the lift and drag coefficients, where the Reynolds and Péclet numbers are 200 and 2,000, respectively. As shown in Fig. **S11**C,D, the algorithm is capable of predicting the lift and drag with good accuracy given both zero-Dirichlet and zero-Neumann boundary conditions on the wall. The zero-Dirichlet boundary condition leads to slightly better predictions as detailed in Table **S4**. This could be attributed to different resolution requirement for the training data. An important observation for these more realistic flow visualizations is that HFM is still agnostic to the geometry and boundary conditions. However, in case the geometry of the wall is explicitly known, one can easily prescribe the no-slip velocity boundary condition at the wall, which leads to slight improvement in the predictions of lift and drag as seen in Fig. **S11**.

[5]The training datasets for these two examples are generated using OpenFoam and are publicly available at `https://github.com/maziarraissi/HFM`.

External 3D flow past a circular cylinder

The previous benchmark example was a two-dimensional (2D) flow, where we could safely neglect the z-coordinate and w-component of the velocity from the input and output variables, respectively. For the flow past a cylinder, if we simply increase the Reynolds number beyond a threshold value of ≈ 185 (*34*), the spanwise velocity w becomes non-zero due to the effect of the so-called "vortex stretching" (*35*). To test the capability of the proposed algorithm in inferring three-dimensional (3D) flow fields, we design another prototype problem in a 3D domain of flow past a finite-size circular cylinder confined between two parallel plates as shown in Fig. **S12**, where similar to the previous example, we set $\text{Re} = \text{Pe} = 100$. Due to three-dimensional geometry and despite the fact that this is a subcritical Reynolds number for the onset of three-dimensionality, the resulted flow is strongly three-dimensional. Downstream of the cylinder, the flow exits to an open region, which causes strong 3D effects in the wake of the cylinder. Hence, we set the training domain in the wake of cylinder.

A representative snapshot of the input data on the concentration field in the wake of the cylinder is plotted in the top left panel of Fig. **S13**. The iso-surfaces for all of the fields are also plotted for comparison between the reference data and the predictions of our algorithm. The algorithm is capable of accurately reconstructing the velocity and pressure fields without having access to any observations of these fields. In particular, no information on the velocity is given on the boundaries of the domain of interest during the training. Qualitative comparison between the predictions of our algorithm and the reference data shows good agreement for 3D flows as well, however, the relative L_2 errors for velocity and pressure fields are slightly higher than the values for the 2D flow predictions as shown in Fig. **S14**. Similar to 2D results, the neural network training lacks sufficient data outside of the training time window, leading to larger errors in the predictions near the beginning and end points of the training time interval.

Stenotic 2D channel flow over an obstacle

We now turn our attention to an important class of flows in confined geometries also known as "internal flows". While measuring average flow velocity and pressure in ducts, pipes and even blood vessels is now a common practice, quantifying the spatial fields, specifically the shear stresses on the boundaries, is still an open problem. We consider internal flow in a 2D channel with a stenosis for which we aim to infer the wall shear stresses (see Fig. **S15**). To make the flow unsteady, we impose a sinusoidal velocity profile at the inlet of the channel. The passive scalar values on the boundaries are set to zero, where the boundaries are assumed to be impenetrable. The presence of the obstacle breaks the symmetry in the flow for which analytical solutions do not exist. Thus, to estimate the velocity field, direct measurements or forward numerical simulations are required.

The predictions of the algorithm for the velocity and pressure fields are shown in Fig. **S16** and are in very good agreement with the reference data. Note that no information for velocity is given on the boundaries of the channel or the obstacle. While this is an idealized problem, it represents some of the complexities encountered in cardiovascular fluid mechanics, where one or multiple coronary arteries are partially blocked by atherosclerotic plaques formed by the lipid accumulation (*36*). Direct measurements of pressure in the vessel are invasive and bear high risk, whereas computational fluid modeling of blood in the coronaries requires precise reconstruction of the geometry as well as knowledge of all the terminal boundary conditions. Furthermore, we present the relative L_2 errors in Fig. **S17**; similar to the previous benchmark problems, slight discrepancies can be observed close to the beginning and end points of the training time window ($t \sim 110.1,\ 130.1$) due to the lack of training data, where the relative L_2 errors in the velocity fields show a significant peak.

Using the predicted velocity fields, we are able to compute shear stresses everywhere in the training domain. Of particular interest are wall shear stresses, which can be computed using the following equations in 2D:

$$\begin{aligned} S_x &= 2\mathrm{Re}^{-1}\left[u_x n_x + \tfrac{1}{2}(v_x + u_y)n_y\right], \\ S_y &= 2\mathrm{Re}^{-1}\left[\tfrac{1}{2}(u_y + v_x)n_x + v_y n_y\right]. \end{aligned} \qquad \textbf{(S6)}$$

Here, $\mathbf{n} = (n_x, n_y)$ is the unit outward normal on the boundary of the domain. Note that to compute the wall shear stress traction vector $\mathbf{S} = (S_x, S_y)$, the gradient of the velocity is required, which can be computed using automatic differentiation. Wall shear stresses are important quantities of interest in many biological processes e.g., in the pathogenesis and progression of vascular diseases that cause aortic aneurysms. We have estimated the temporal wall shear stress magnitudes, WSS $=\sqrt{S_x^2 + S_y^2}$, acting on the lower wall using the predictions of neural networks for the 2D channel flow over the obstacle, and plotted them against the results from the spectral/hp element solver in Fig. S18; the results show excellent agreement between the predictions and numerical estimations. Note that the numerical estimations suffer from a slight aliasing effect (noisy oscillations in the x direction) for which dealiasing is required to retrieve a smooth distribution of wall shear stresses. Interestingly, the neural networks prediction is smooth as the velocity gradients of a parameterized velocity field are taken analytically. The proposed inverse approach to infer the velocity and pressure fields with the use of a passive scalar, and hence the wall shear stress, offers a promising alternative to conventional methods. Here, the passive scalar could be the bolus dye that is typically injected to the blood stream for the purpose of blood flow monitoring and medical imaging.

Lastly, similar to the external flow problem past the cylinder, the algorithm is able to infer the Reynolds and Péclet numbers as free parameters. The inferred values of these two parameters are given in Table S3, which shows an excellent agreement with the reference values. Note that unlike the flow past the cylinder, the Re and Pe numbers are not the same.

3D Intracranial aneurysm

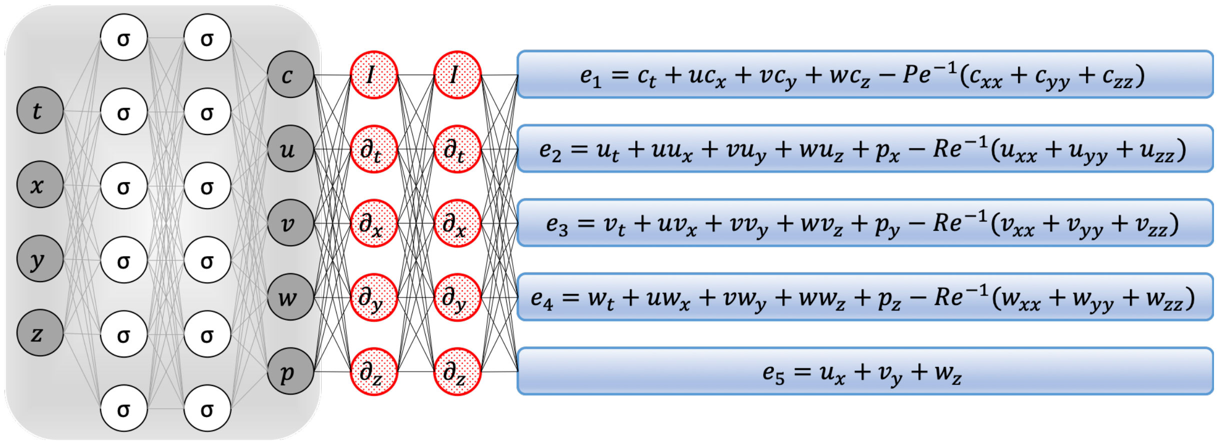
To further illustrate the implications of the *Navier-Stokes informed neural networks* in addressing real-world problems, we consider 3D physiologic blood flow in a realistic intracranial aneurysm (ICA) shown in Figs. S19 and 3A. Reference concentration fields are generated numerically using realistic boundary conditions representing a physiologic flow waveform at the inlet along with a uniform concentration for the passive scalar. The strength of the proposed algorithm is its ability to stay agnostic with respect to the geometry and boundary conditions. Consequently, we first crop the aneurysm sac out from the rest of geometry, and then use only the scalar data within the ICA sac (right panel of Figs. S19 and 3B) for training where no information is used for the boundary conditions. The reference and predicted concentration, velocity and pressure fields within the ICA sac at a sample time instant are then interpolated on two separate planes perpendicular to y- and z-axis, which are shown in Fig. 3D. We observe acceptable agreement between the reference and predicted fields given the complexity of the flow field. Furthermore, the relative L_2 errors plotted in Fig. S20 show a significant drop in the prediction errors away from the beginning and end points of the training time window, which justifies an accurate analysis of model predictions in the time interval $94.09 \leq t \leq 109.09$. Furthermore, we have estimated the wall shear stress components on the surface of the aneurysm sac for this complicated geometry. The computations can be performed fairly easily so long as the outward wall normals are known and given to the algorithm. The reference and predicted wall shear stresses (given by the components of the traction vector) as well as the pressure field acting on the aneurysm wall are shown in Fig. S21. The wall traction vector is defined by $\mathbf{S} = \boldsymbol{\tau} \cdot \mathbf{n}|_{wall}$, where $\boldsymbol{\tau}$ is the shear stress tensor field and $\mathbf{n}$ is the unit outward normal to the wall. The formulas to compute the wall traction vector in 3D are a straightforward generalization of Eq. S6 in 2D.

Remarks

The emphasis of this study was to demonstrate, through several prototypical and realistic examples, the ability of the current algorithm to infer the hidden states of the system from partial knowledge of some relevant quantities by leveraging the known underlying dynamics of the system. We have verified that with the use of governing physical laws, our algorithm is able to make accurate predictions for complex 2D/3D flows. One possible limitation of the current study is the use of synthetically generated data on the passive scalar, which are relatively noiseless and clean compared to the realistic measurements. To address this issue, we have carried out extensive systematic studies with respect to the spatio-temporal resolution of the training data as well as noise levels. Whereas 3D reconstruction of a passive scalar from a stack of 2D projections is possible, it is not a common practice in the clinical and industrial settings due to the technological complexities as well as computational and processing costs. Thus, extracting information from 2D images directly by taking into account the angle of projections seems to be a more realistic approach. Furthermore, different imaging modalities (e.g., magnetic resonance vs. computed tomography angiography of bolus dye in coronary arteries) have different spatial/temporal resolutions. In the first benchmark problem, we verified the robustness of the HFM for low temporal resolution e.g., five snapshots per shedding cycle. However, if the number of snapshots is inadequate, a viable strategy is to use a time-discrete HFM by employing a high-order time-stepping scheme such as implicit Runge-Kutta with 100 stages (c.f. (*5*)) for integrating the governing equations in time using the snapshots as initial and terminal conditions. These are certainly important factors that have to be taken into account moving forward using the proposed algorithm for real-world fluid mechanics applications, and will be addressed carefully in future implementations.

Assuming the geometry to be known, to arrive at similar results as the ones presented in the current work, one needs to solve significantly more expensive optimization problems using conventional PDE solvers (e.g., finite elements, finite volumes and etc.). The corresponding optimization problems normally involve some form of “parametrized” initial and boundary conditions, appropriate loss functions, and multiple runs of the forward simulations. In this setting, one could easily end up with very high-dimensional optimization problems that require either backpropagating through the computational solvers or “Bayesian” optimization techniques for the surrogate models (e.g., Gaussian processes). If the geometry is further assumed to be unknown (as is the case in this work), then its parametrization requires grid regeneration, which makes the approach almost impractical.

Lastly, in this work we have been operating under the assumption of Newtonian and incompressible fluid flow governed by the Navier-Stokes equations. However, the proposed algorithm can also be used when the underlying physics is non-Newtonian, compressible, or partially known. This in fact is one of the advantages of the algorithm in which other unknown parameters such as the Reynolds and Péclet numbers can be inferred in addition to the velocity and pressure fields. When the fluid is non-Newtonian (e.g., blood flow in small vessels), one can encode the momentum equations, where the “constitutive” law for the fluid’s stress-strain relationship is unknown, into a physics-informed deep learning algorithm. Having data on the velocity (or even the passive scalar), it may be possible to learn the constitutive law as well. Moreover, it may be also possible to apply HFM to high-speed aerodynamics inferring the velocity and pressure fields from Schlieren-type images representing density gradient along a certain direction.



t
x
y
z
σ
c
u
v
w
p
I
∂_t
∂_x
∂_y
∂_z
$e_1 = c_t + uc_x + vc_y + wc_z - Pe^{-1}(c_{xx} + c_{yy} + c_{zz})$
$e_2 = u_t + uu_x + vu_y + wu_z + p_x - Re^{-1}(u_{xx} + u_{yy} + u_{zz})$
$e_3 = v_t + uv_x + vv_y + wv_z + p_y - Re^{-1}(v_{xx} + v_{yy} + v_{zz})$
$e_4 = w_t + uw_x + vw_y + ww_z + p_z - Re^{-1}(w_{xx} + w_{yy} + w_{zz})$
$e_5 = u_x + v_y + w_z$

Fig. S1. Navier-Stokes informed neural networks. A fully connected (physics-uninformed) feed-forward neural network, with 10 hidden layers and 50 neurons per hidden layer per output variable (i.e., $5 \times 50 = 250$ neurons per hidden layer), takes the input variables t, x, y, z and outputs c, u, v, w, and p. As for the activation functions σ, we use the *swish* activation function (*25*). For illustration purposes only, the network depicted in this figure comprises of 2 hidden layers and 6 neurons per hidden layers. We employ automatic differentiation to obtain the required derivatives to compute the residual (physics-informed) networks e_1, e_2, e_3, e_4, and e_5. If a term does not appear in the blue boxes (e.g., u_{xy} or u_{tt}), its coefficient is assumed to be zero. It is worth emphasizing that unless the coefficient in front of a term is non-zero, that term does not appear in the "compiled" computational graph and will not contribute to the computational cost of a feed forward evaluation of the resulting network. The total loss function is composed of the regression loss of the passive scalar c on the training data, and the loss imposed by the differential equations $e_1 - e_5$. Here, I denotes the identity operator and the differential operators $\partial_t, \partial_x, \partial_y$, and ∂_z are computed using automatic differentiation, and can be thought of as "activation operators". It is worth noting that the aforementioned differential operators are non-trivial in the sense that they almost double the length of the network each time they are applied due to the need for the chain rule operations necessary to arrive at the input variables (i.e., t, x, y, and z). Given that the convection-diffusion equations involve second order derivatives, the depth of the physics-informed neural networks e_1, e_2, e_3, e_4 is practically $40 = 2 \times (2 \times 10)$ whereas the depth of e_5 is $20 = 2 \times 10$. Moreover, the gradients of the loss function are backpropogated through the entire network to train the parameters using the Adam optimizer.

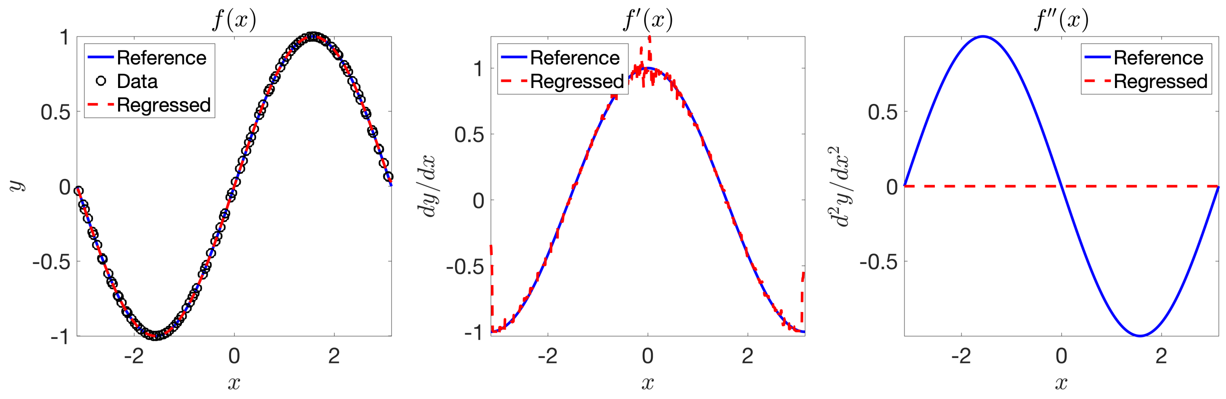


Fig. S2. Regressing data generated from $f(x) = \sin(x)$ using the ReLU activation function. A neural network with ReLU activation can accurately approximate $f(x)$ (left panel), while it leads to a non-smooth regression for the first derivative of f (middle panel) and a vanishing second derivative of f (right panel).

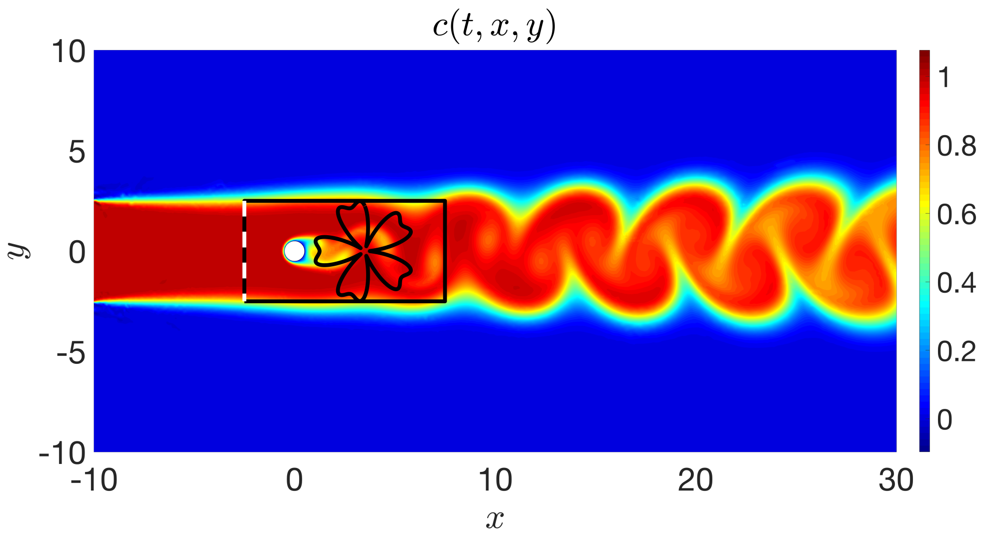


Fig. S3. Problem setup for 2D flow past a circular cylinder. Simulation domain in which data on the concentration field is generated. We have assumed a uniform free stream velocity profile at the left boundary, a zero pressure outflow condition imposed at the right boundary located $30D$ $(D = 1)$ downstream of the cylinder, and periodicity for the top and bottom boundaries of the $[-10, 30] \times [-10, 10]$ domain. The passive scalar is injected at the inlet from [-2.5, 2.5]. A completely arbitrary training domain in the shape of a flower is depicted in the wake of the cylinder. No information on the velocity is given for this training domain. Another training domain is also shown by a rectangle that includes the cylinder. Information on the velocity is only given on the left boundary of the rectangular domain shown by the dashed line.

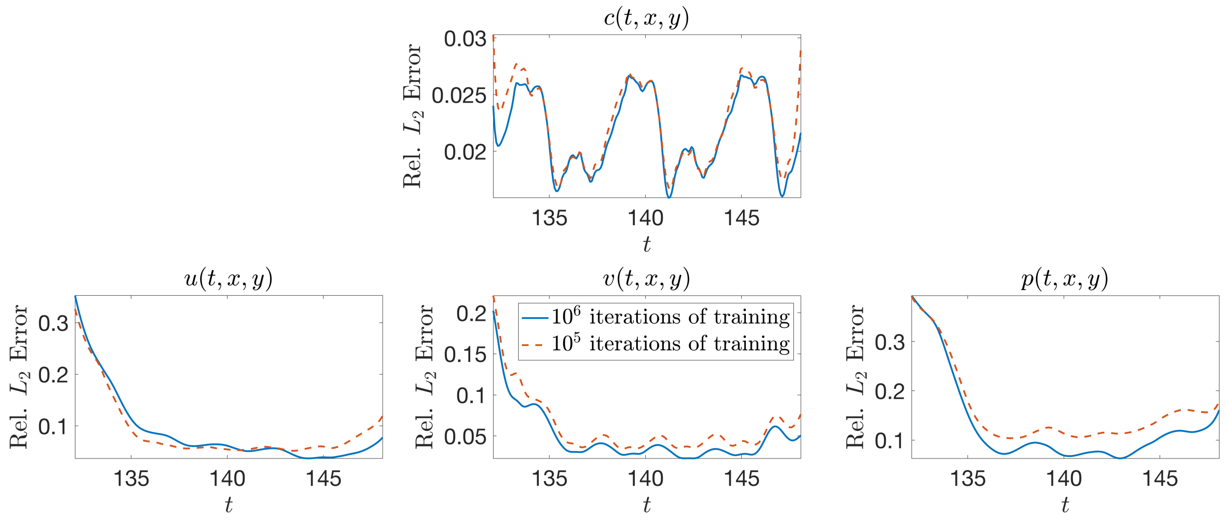


Fig. S4. Relative L_2 errors for 2D flow past a circular cylinder with arbitrary training domain. Relative L_2 errors between the model output and reference fields computed over the full time window. The results are shown for the most refined input data, where 31 million data points scattered in space and time are used both to regress the concentration field and enforce the governing conservation equations.

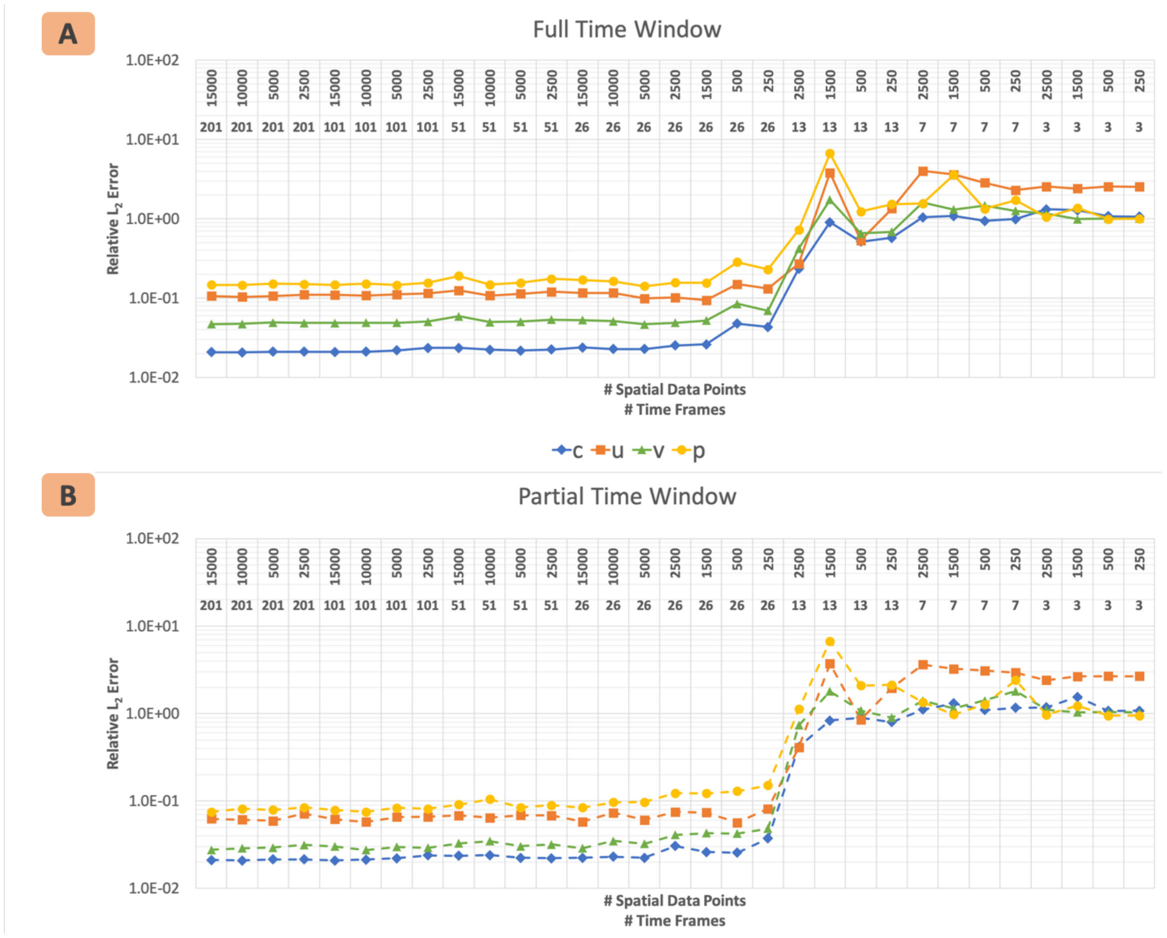


Fig. S5. Relative L_2 errors for 2D flow past a circular cylinder with arbitrary training domain. Results are shown for various spatio-temporal resolutions of the input concentration data. **A:** Relative L_2 error computed over the full time window $t = 132.08 - 148.08$. **B:** Relative L_2 error computed over partial time window $t = 137.08 - 142.08$. The spatial resolution is considered between 250-15,000 data points, whereas the time resolution is between 3-201 time frames.

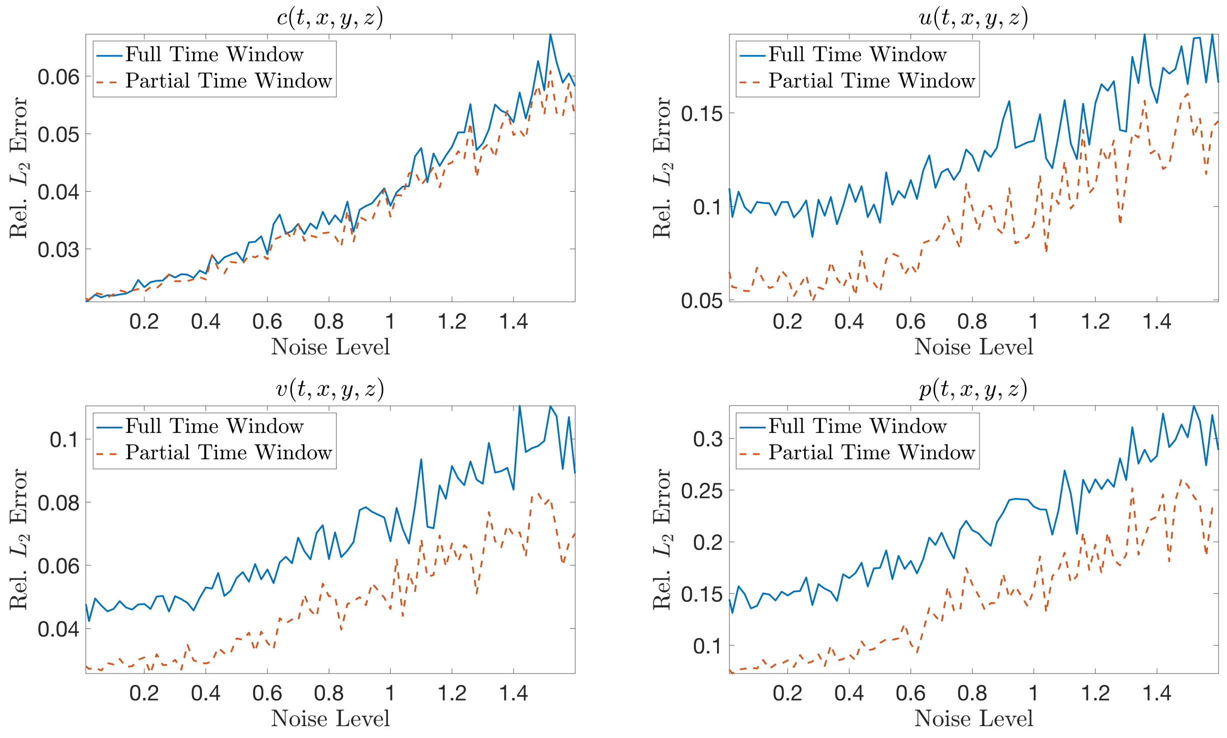


Fig. S6. Relative L_2 errors of model output using corrupted concentration data with additive Gaussian noise. Relative L_2 errors are plotted for concentration, velocity and pressure fields as a function of the noise level. The problem under investigation is the 2D flow past a circular cylinder with arbitrary training domain. The level of the noise is defined as the level of distortion in the innate variations (i.e., standard deviation) of data given by Eq. S5.

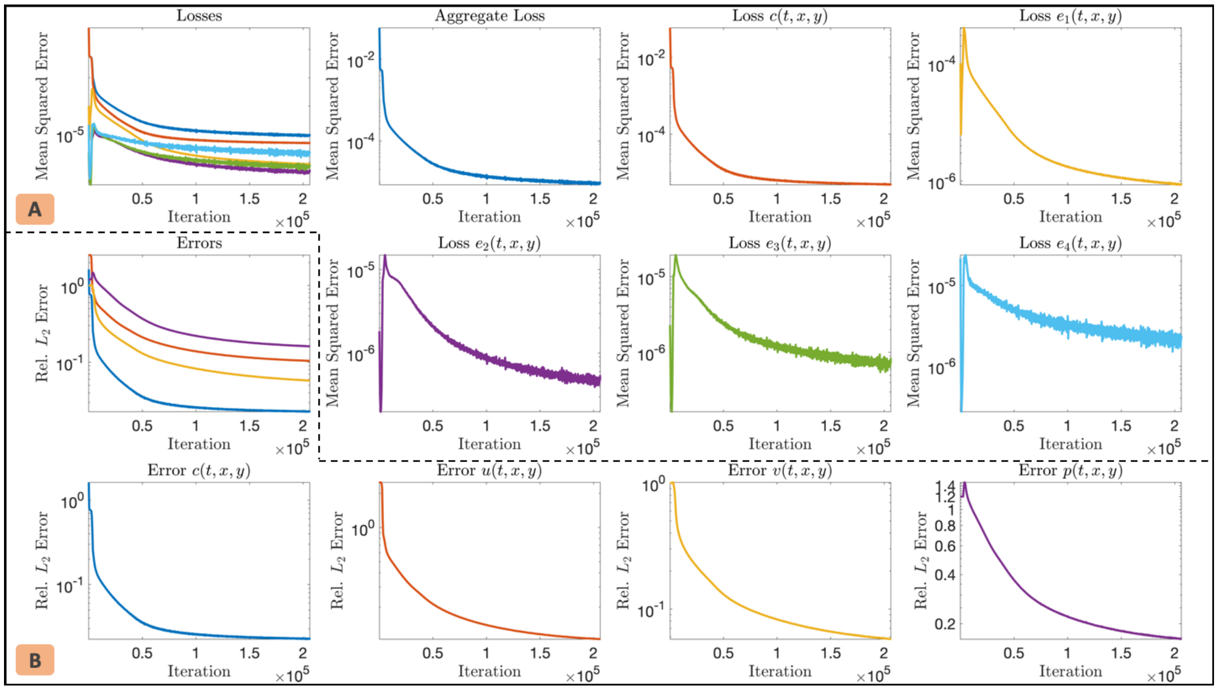


Fig. S7. History of losses and relative L_2 errors for 2D flow past a circular cylinder with arbitrary training domain. A: History of the aggregate loss (loss of data plus the enforced equations) as well as each individual loss on data $c(t,x,y)$ and the enforced equations $e_1 - e_4$ are plotted with respect to the number of iterations. The mean squared loss function is defined in Eq. 1. **B**: History of the relative L_2 error between the model output and the reference concentration, velocity and pressure fields are plotted with respect to the number of iterations. The relative L_2 error is defined by Eq. S4. In this figure, to filter out the oscillations in the time series for the loss values and the relative L_2 errors, we employ a centered moving average by sliding a window of length 500 iterations along the corresponding time series.

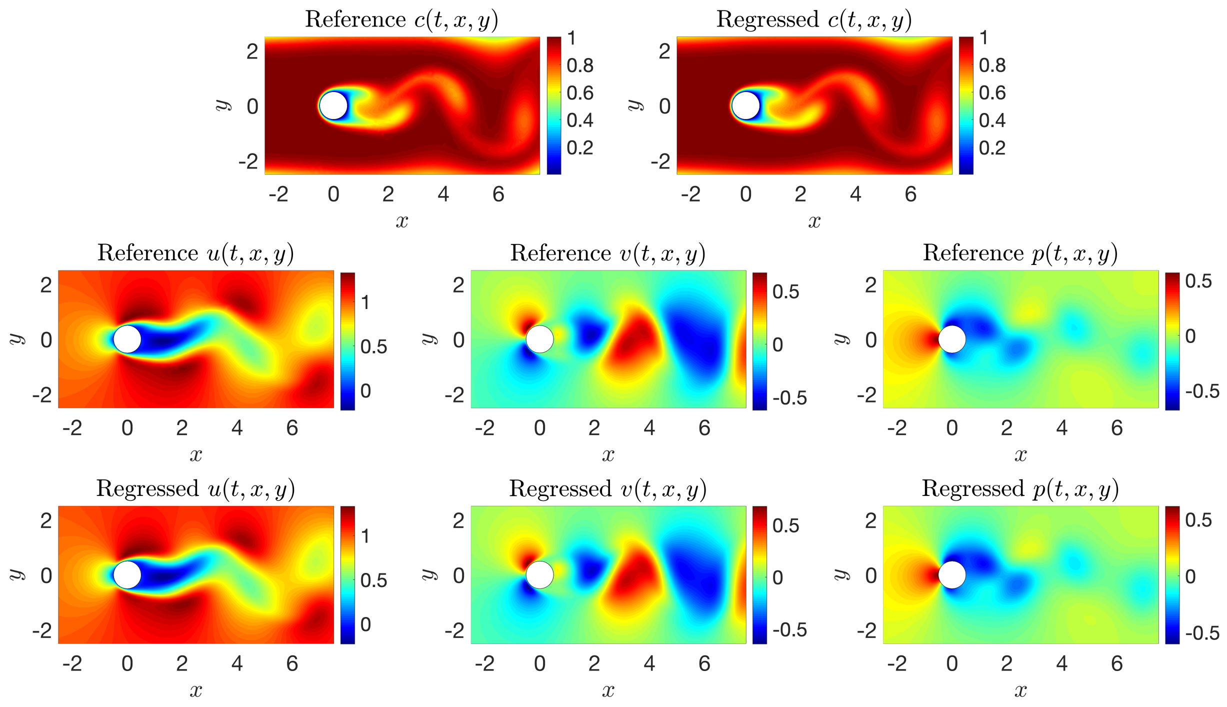


Fig. S8. 2D flow past a circular cylinder with rectangular training domain. A representative snapshot of the input data on the concentration of the passive scalar is shown in the top left panel of this figure, where on the right the same concentration field is regressed by our algorithm. The algorithm is capable of accurately regressing the velocity u, v and the pressure p fields shown in the third row. The reference velocity and pressure fields at the same point in time are plotted for comparison in the second row. Note that the pressure is only identifiable up to an additive constant since this is an incompressible flow and only the spatial derivatives of the pressure appear in the momentum equations.

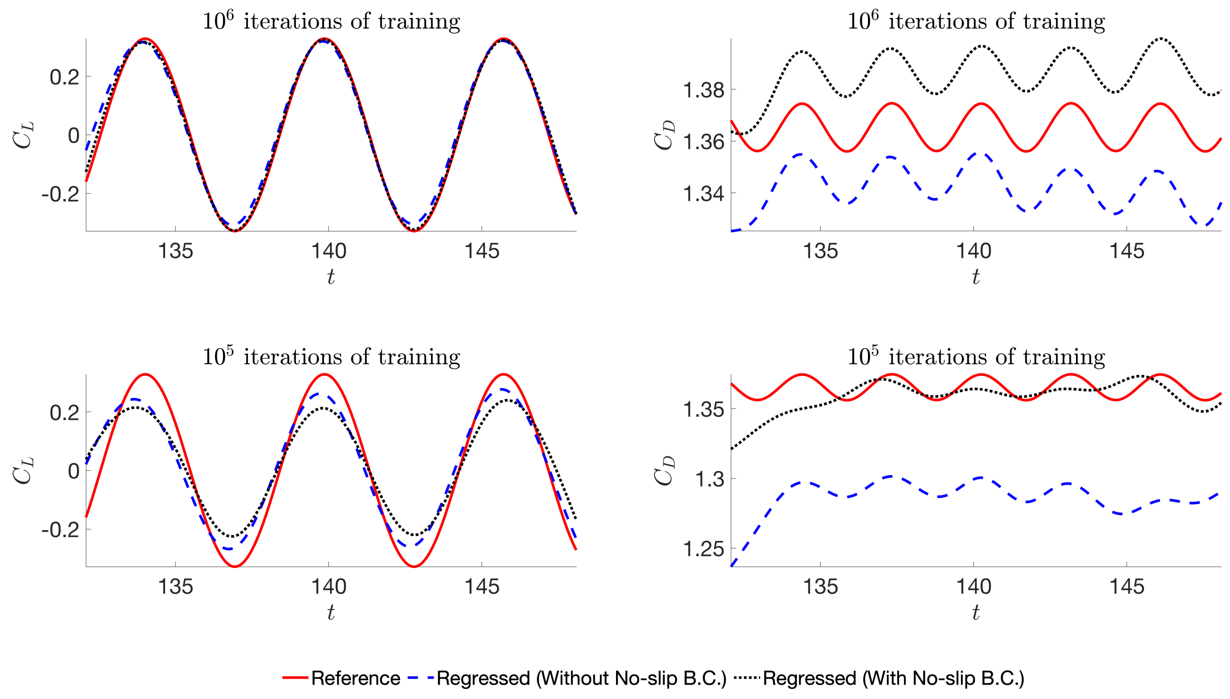


Fig. S9. Predicted lift and drag acting on a cylinder after 10^6 and 10^5 iterations of training. Prediction vs. reference lift coefficient C_L and drag coefficient C_D over the time window of 2.5 vortex shedding cycles. The no-slip boundary condition on the cylinder wall is prescribed during the training for comparison, and the results are shown in black dotted lines.

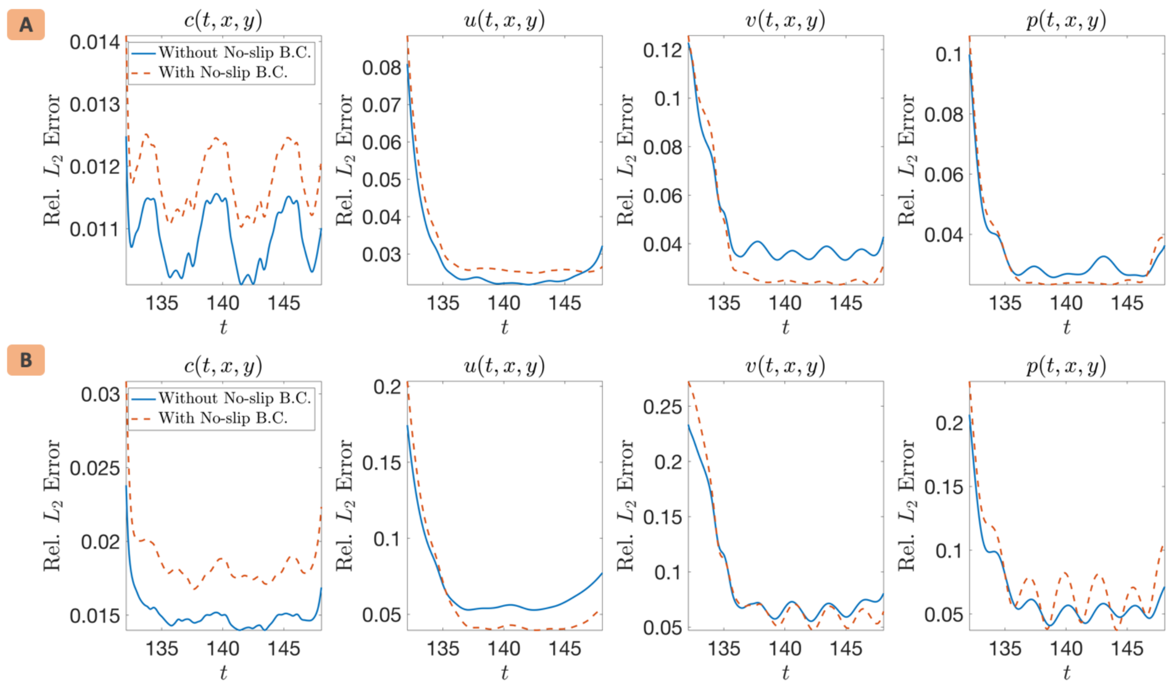


Fig. S10. Relative L_2 errors for 2D flow past a circular cylinder with rectangular training domain. Relative L_2 errors between the model output and reference fields computed over the full time window after **A:** 10^6 and **B:** 10^5 iterations of training. The results are shown for the most refined input data, where 2.8 million data points scattered in space and time are used both to regress the concentration field and enforce the governing conservation equations. Furthermore, in each panel the errors are shown for two scenarios: no-slip boundary condition on the cylinder wall is not imposed (solid blue line) vs. prescribed no-slip boundary condition (dashed red line).

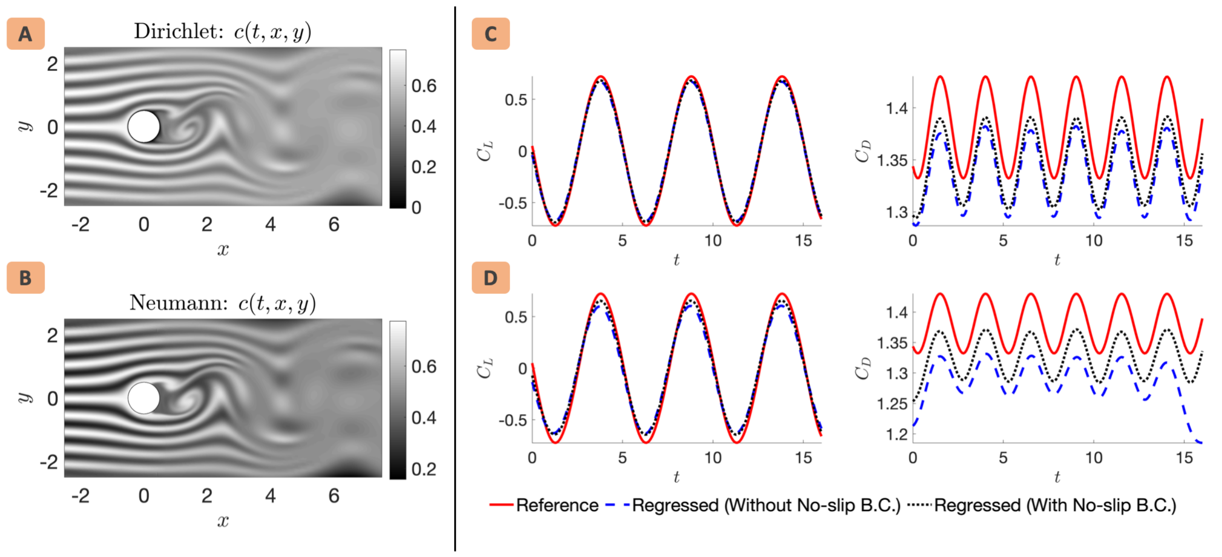


Fig. S11. Comparison of the effect of Dirichlet and Neumann boundary conditions on the predicted lift and drag acting on a cylinder using streaklines as training data. This example is inspired by smoke visualizations of flow around an airfoil at an angle of attack, see e.g., Fig. 72 in (*33*). Flow streaklines around a circular cylinder visualized by the passive scalar using **A:** zero-Dirichlet; and **B:** zero-Neumann boundary condition. Here, Re = 200 and Pe = 2,000. A sinusoidal function in the form of $-2.5 \leq y \leq 2.5: \; c(y) = [1 + \cos(4\pi y)]/2$ at the inlet was used to produce these streaklines. These data are generated using OpenFoam. Prediction vs. reference lift coefficient C_L and drag coefficient C_D over the time window of 2.5 vortex shedding cycles, where the data are generated using **C:** zero-Dirichlet; and **D:** zero-Neumann boundary condition. The no-slip boundary condition on the cylinder wall can also be prescribed during the training with the results shown in black dotted lines.

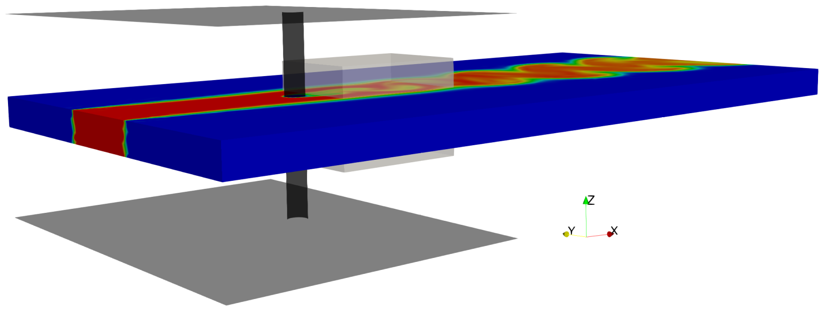


Fig. S12. Problem setup for 3D flow past a circular cylinder. 3D sketch for the prototypical problem of flow past a circular cylinder of diameter $D = 1$. Two parallel planes are located $10D$ apart along the $z-$axis. Flow is bounded between the plates for $10D$ along $x-$axis. Periodic boundary conditions are prescribed parallel to the xz plane at $y = -10$ and 10, and parallel to the xy plane at $z = 0$ and 10 starting from the location where the walls end. All the physical boundaries are shown in black, whereas the gray box located behind the cylinder shows the domain of interest where model training is performed. Zero-Neumann boundary conditions are imposed for velocities and concentration along with zero pressure at the outflow located $30D$ downstream of the cylinder. A uniform $U = 1$ is imposed at the inlet located $8D$ upstream of the cylinder. Passive scalar is injected at the inlet through a finite region $[-2.5, 2.5] \times [0, 10]$. Zero-velocity and concentration boundary conditions are imposed on each physical boundary. Note that only concentration data in the training (gray box) domain are given to the neural networks.

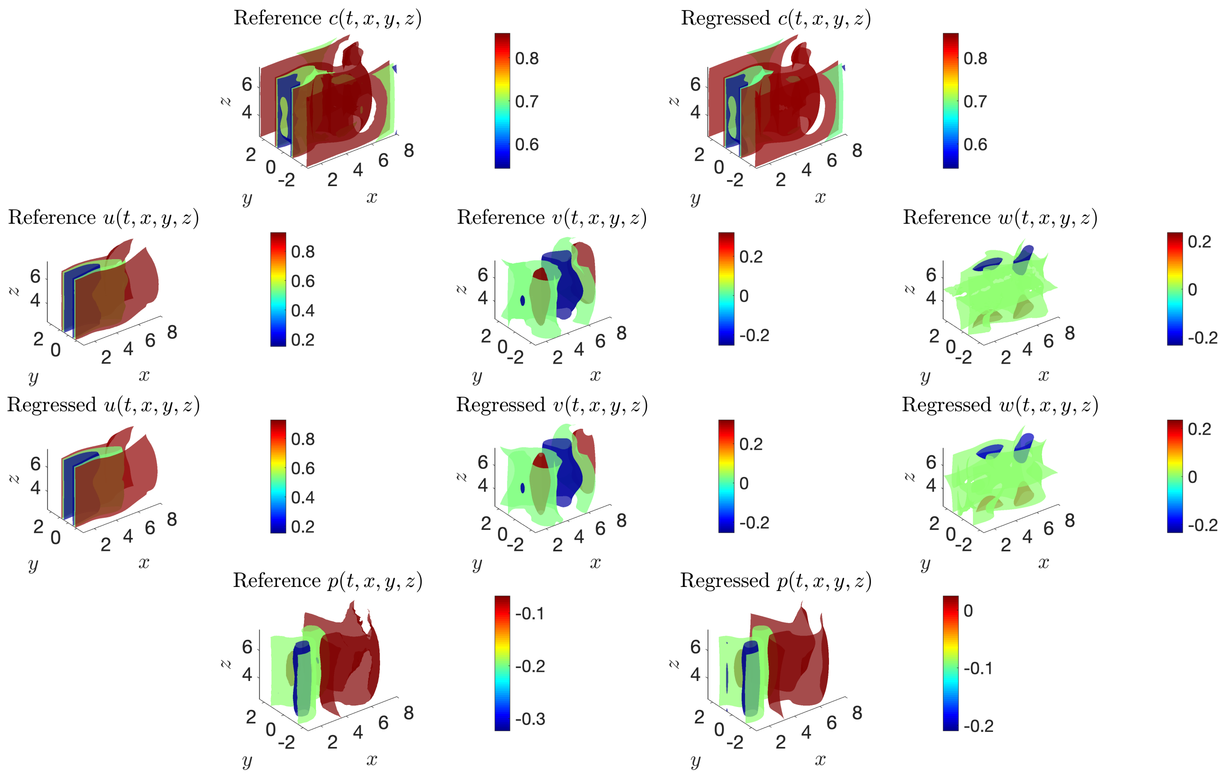


Fig. S13. 3D flow past a circular cylinder. The iso-surfaces of reference data and the regressed concentration of the passive scalar are shown in top row for a representative time instant within the selected training domain. Using the information on the concentration only, the velocity fields u, v, w are inferred (shown in the third row) and are compared with the reference data in the second row. In addition, the reference and regressed pressure p fields are plotted in the last row. Note the range of contour levels are set equal between the reference and regressed iso-surfaces of concentration and velocity components for better comparison.

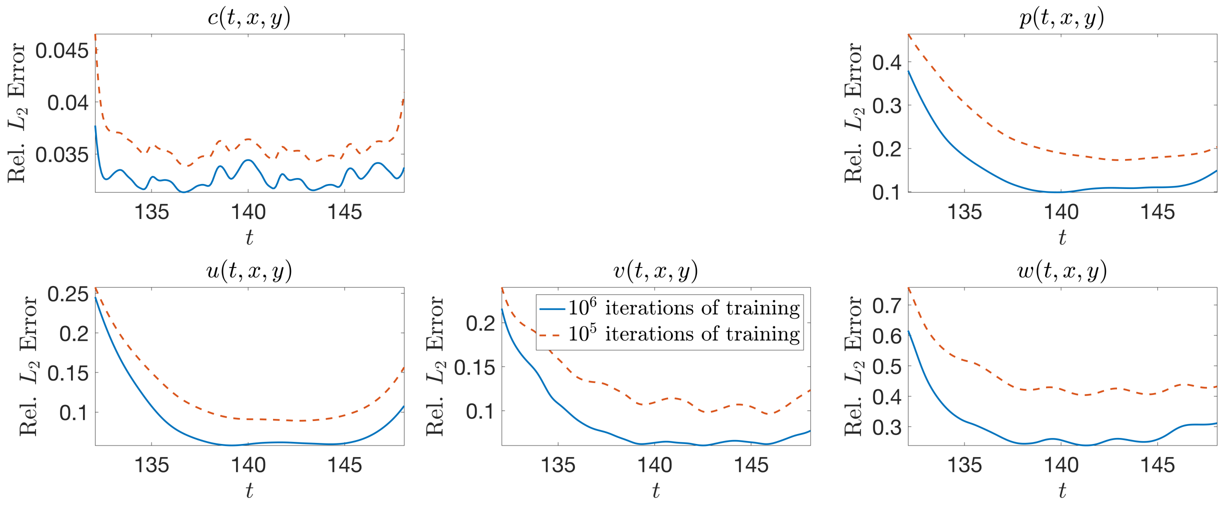


Fig. S14. Relative L_2 errors for 3D flow past a circular cylinder. Relative L_2 errors between the model output and reference fields computed over the full time window. 37 million data points scattered within the training box and in time are used for both regressing the concentration field and enforcing the corresponding partial differential equations.

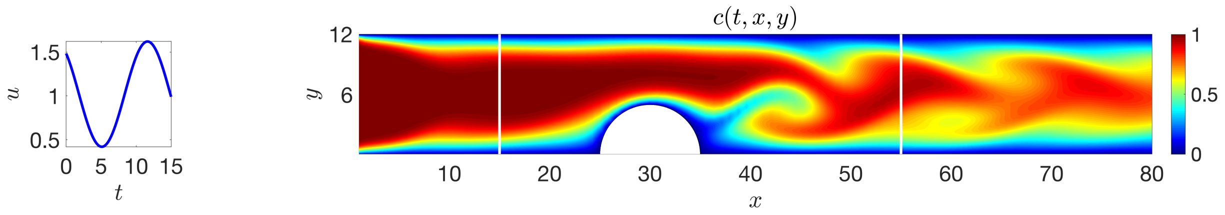


Fig. S15. Problem setup for 2D channel flow over an obstacle. Contours of the concentration field within a 2D channel at a representative time instant are shown in the right panel. A sinusoidal velocity profile $u(t)$ ($v = 0$) is imposed at the inlet (shown on the left), and a uniform concentration $c = 1$ for the passive scalar is injected to the channel. At the outlet, a zero-Neumann boundary condition for the velocity and concentration is considered, while the pressure is set to zero. Furthermore, the velocity and concentration on the walls are set to zero. The flow Re $= \bar{U}H/\nu = 60$ is calculated based on the mean inflow velocity $\bar{U} = 1$ and channel height $H = 12$, whereas we choose a smaller diffusion constant for the passive scalar leading to higher Pe $= \bar{U}H/\kappa = 180$. The training domain (white rectangle) is considered to contain the obstacle with the upper and lower boundaries being the physical wall boundaries.

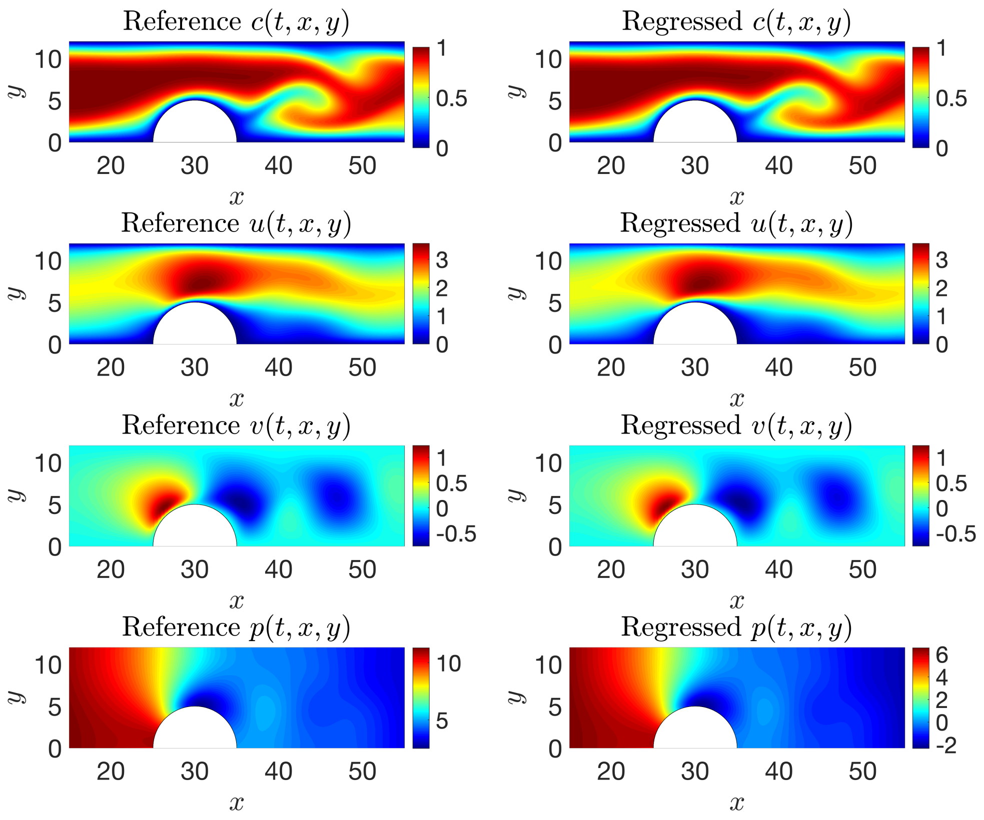


Fig. S16. 2D channel flow over an obstacle. The outputs from the neural networks include the regressed concentration field based on the predictions of HFM, and the regressed velocity u, v and pressure p fields shown on the right column. Reference concentration, velocity and pressure fields are plotted for comparison on the left column. Note that the pressure is off by an additive constant since this is an incompressible flow and only the spatial derivatives of the pressure appear in the momentum equations.

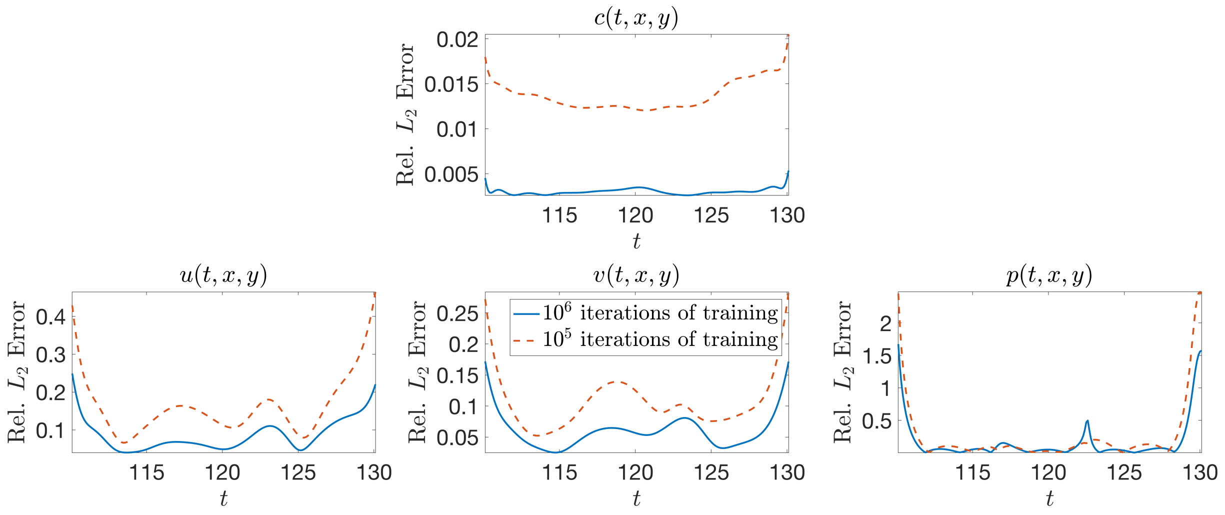


Fig. S17. Relative L_2 errors for 2D channel flow over an obstacle. Relative L_2 errors between the model output and reference fields computed over the full time window. 10 million data points scattered within the training domain and in time are used for both regressing the concentration field and enforcing the corresponding partial differential equations.

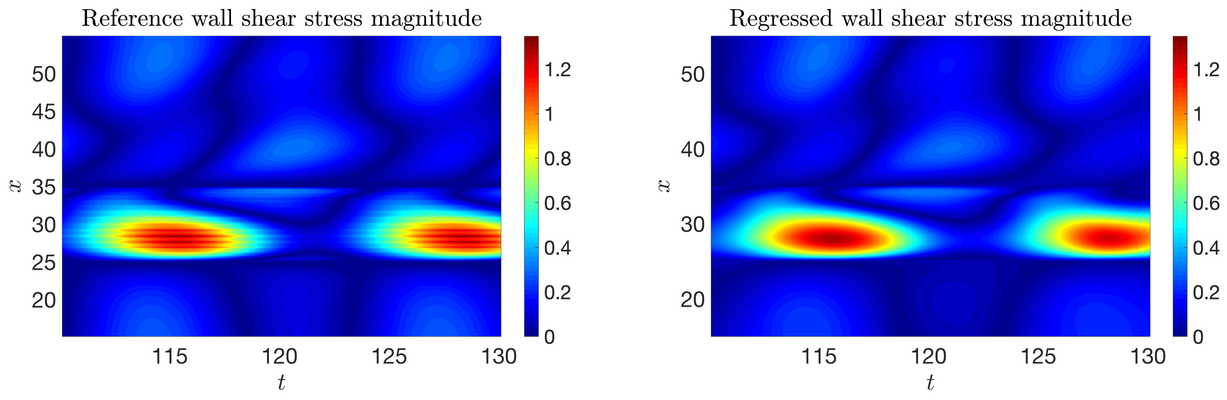


Fig. S18. Wall shear stresses for 2D channel flow over an obstacle. Regressed wall shear stress magnitude as a function of space and time acting on the lower side of the training domain including the circular arc is shown in the right panel. The reference wall shear stress distribution is shown in the left panel for comparison.

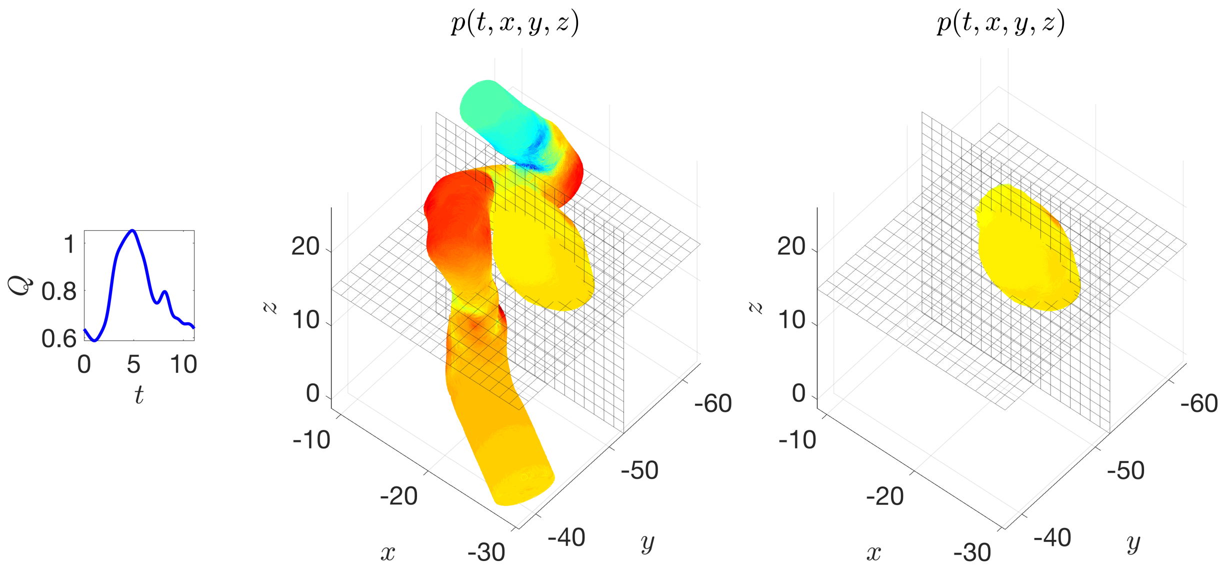


Fig. S19. Problem setup for 3D intracranial aneurysm. The middle panel shows the simulation domain and pressure field at a time instant, whereas on the right the training domain containing only the ICA sac is shown. Two perpendicular planes have been used to interpolate the exact data and the predicted ones for plotting 2D contours in Fig. 3. A physiologic flow waveform $Q(t)$ (shown in the inset figure) is prescribed at the inlet along with the uniform concentration for the passive scalar. At the outlet, zero-Neumann boundary conditions are imposed for velocities and concentration, whereas a "Windkessel" type boundary condition is used for the pressure to represent the truncated geometry downstream (*37*). The Reynolds and Péclet numbers are estimated based on the mean velocity and lumen diameter at the inlet. Using the kinematic viscosity of blood and assuming a diffusivity constant κ for the passive scalar equal to the viscosity, we obtain $\text{Re} = \text{Pe} = 98.2$.

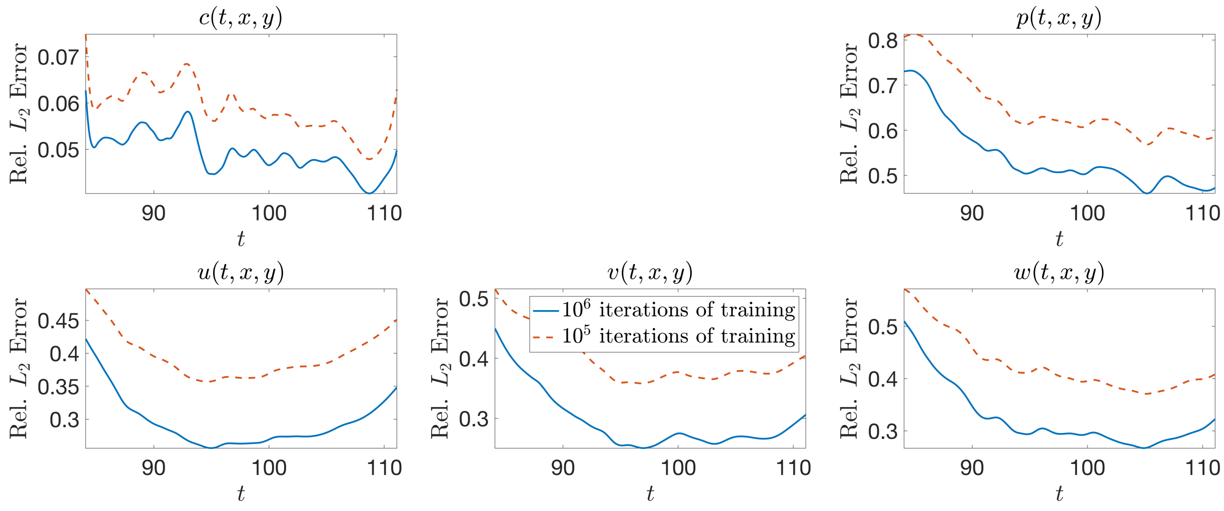


Fig. S20. Relative L_2 errors for 3D intracranial aneurysm. Relative L_2 errors between the model output and reference fields computed over the full time window. 29 million data points scattered within the aneurysm sac and in time are used for both regressing the concentration field and enforcing the corresponding partial differential equations.

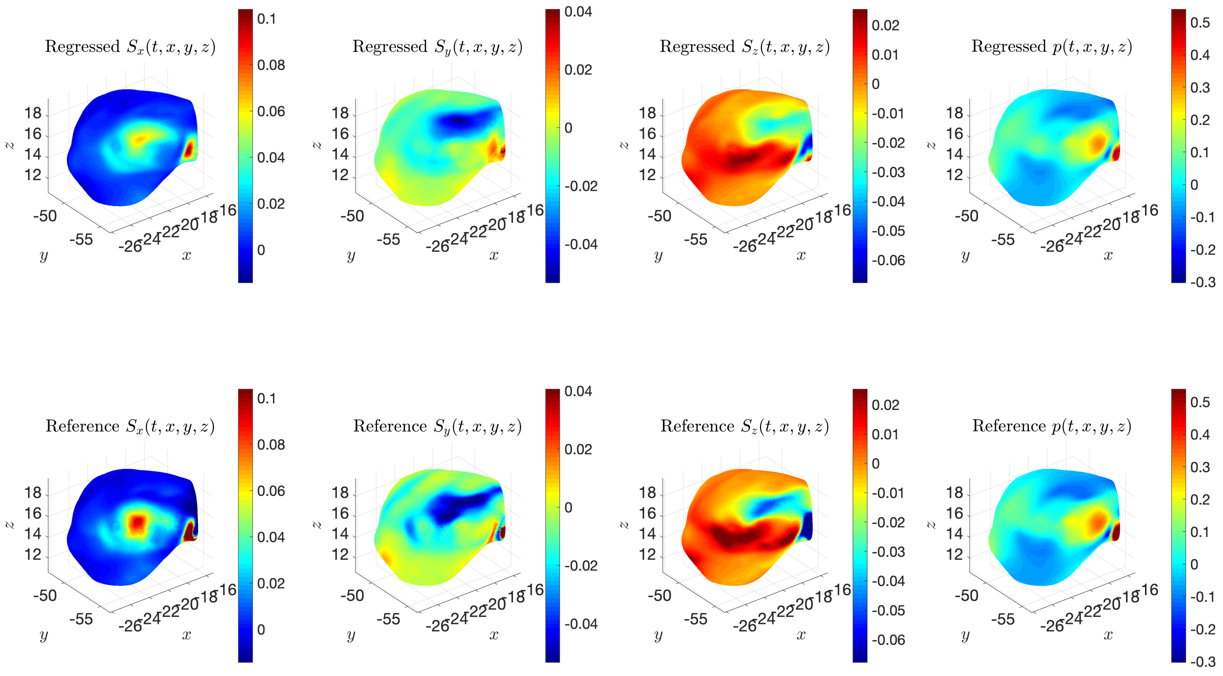


Fig. S21. Wall shear stresses for the 3D intracranial aneurysm. Wall shear stresses are shown as the components of the traction vector $\mathbf{S} = (S_x, S_y, S_z)$ for comparison between the model output and the reference fields. Pressure fields p acting on the aneurysm wall are also plotted. The dataset for this example is generated using OpenFoam and is publicly available at https://github.com/maziarraissi/HFM.

Table S1. Forward Simulations. Concentration data are produced by forward numerical simulations for the benchmark problems investigated in this work using the open-source spectral/hp element solvers (Nektar/Nektar++), where DoF is the degrees of freedom, Δt is the timestep size and N is the number of timesteps to reach the final solution.

	# Processors	DoF	Δt	N
Da Vinci's drawing in Fig. **1** (Nektar++)	144	22,476	0.001	5,160,000
Cylinder 2D in Fig. **2** (Nektar)	384	1,023,400	0.0008	185,100
Cylinder 3D in Fig. **S12** (Nektar)	1248	2,722,640	0.0008	185,100
Stenosis 2D in Fig. **S15** (Nektar)	240	472,745	0.001	130,100
Aneurysm 3D in Fig. **3** (Nektar)	576	813,350	0.00075	148,120

Table S2. 2D flow past a circular cylinder. Inferred Reynolds and Péclet numbers considered as free parameters of the model after 10^6 and 10^5 iterations of training.

		10^6 iterations of training		10^5 iterations of training	
	Reference	Inferred	Rel. Error	Inferred	Rel. Error
Pe	100	93.41	6.59%	91.07	8.93%
Re	100	93.16	6.84%	88.54	11.46%

Table S3. 2D channel flow over an obstacle. Inferred Reynolds and Péclet numbers considered as free parameters of the model after 10^6 and 10^5 iterations of training.

		10^6 iterations of training		10^5 iterations of training	
	Reference	Inferred	Rel. Error	Inferred	Rel. Error
Pe	180	180.40	0.22%	184.00	2.22%
Re	60	58.26	2.90%	53.54	10.76%

Table S4. Relative L_2 errors for 2D flow past a circular cylinder with streaklines as data. Different wall boundary conditions are used for comparison. The errors are between the model output and reference fields. They are computed within the training domain (rectangular domain shown in Fig. S3) and the corresponding time window for c, u, v and p.

	Relative L_2 errors (%)			
	c	u	v	p
$c\|_{wall} = 0$ with (without) prescribed no-slip velocity condition	3.84 (3.89)	4.69 (4.79)	8.76 (8.45)	4.99 (5.13)
$\partial c/\partial n\|_{wall} = 0$ with (without) prescribed no-slip velocity condition	4.59 (4.68)	7.22 (9.6)	9.61 (12.26)	6.58 (8.27)

Movie S1. Flow visualization of passive scalar transport in the aneurysm generated by direct numerical simulation.

Movie S2. Comparison of the regressed flow dynamics vs. the reference flow fields; shown here by the velocity streamlines that are colored by pressure.

References and Notes

1. J. Westerweel, D. Dabiri, M. Gharib, The effect of a discrete window offset on the accuracy of cross-correlation analysis of digital PIV recordings. *Exp. Fluids* **23**, 20–28 (1997). doi:10.1007/s003480050082

2. F. Pereira, J. Lu, E. Castano-Graff, M. Gharib, Microscale 3D flow mapping with μDDPIV. *Exp. Fluids* **42**, 589–599 (2007). doi:10.1007/s00348-007-0267-5

3. M. Shattuck, R. Behringer, G. Johnson, J. Georgiadis, Convection and flow in porous media. Part 1. Visualization by magnetic resonance imaging. *J. Fluid Mech.* **332**, 215–245 (1997). doi:10.1017/S0022112096003990

4. G. K. Batchelor, *An Introduction to Fluid Dynamics* (Cambridge University Press, 2000).

5. M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707 (2019). doi:10.1016/j.jcp.2018.10.045

6. G. E. Karniadakis, S. Sherwin, *Spectral/hp Element Methods for Computational Fluid Dynamics* (Oxford University Press, 2nd Edition, 2013).

7. H. Baek, G. E. Karniadakis, A convergence study of a new partitioned fluid–structure interaction algorithm based on fictitious mass and damping. *J. Comput. Phys.* **231**, 629–652 (2012). doi:10.1016/j.jcp.2011.09.025

8. J. B. Barlow, W. H. Rae Jr., A. Pope, *INCAS Bull.* **7**, 133 (2015).

9. M. Koochesfahani, P. Dimotakis, Laser-induced fluorescence measurements of mixed fluid concentrationin a liquid plane shear layer. *AIAA J.* **23**, 1700–1707 (1985). doi:10.2514/3.9154

10. J. Crimaldi, Planar laser induced fluorescence in aqueous flows. *Exp. Fluids* **44**, 851–863 (2008). doi:10.1007/s00348-008-0496-2

11. S. Voros, S. Rinehart, Z. Qian, P. Joshi, G. Vazquez, C. Fischer, P. Belur, E. Hulten, T. C. Villines, Coronary atherosclerosis imaging by coronary CT angiography: Current status, correlation with intravascular interrogation and meta-analysis. *JACC Cardiovasc. Imaging* **4**, 537–548 (2011). doi:10.1016/j.jcmg.2011.03.006 Medline

12. M. Wintermark, M. Reichhart, J.-P. Thiran, P. Maeder, M. Chalaron, P. Schnyder, J. Bogousslavsky, R. Meuli, Prognostic accuracy of cerebral blood flow measurement by perfusion computed tomography, at the time of emergency room admission, in acute stroke patients. *Ann. Neurol.* **51**, 417–432 (2002). doi:10.1002/ana.10136 Medline

13. A. R. Aron, T. E. Behrens, S. Smith, M. J. Frank, R. A. Poldrack, Triangulating a cognitive control network using diffusion-weighted magnetic resonance imaging (MRI) and functional MRI. *J. Neurosci.* **27**, 3743–3752 (2007). doi:10.1523/JNEUROSCI.0519-07.2007 Medline

14. A. M. Shaaban, A. J. Duerinckx, Wall shear stress and early atherosclerosis: A review. *AJR Am. J. Roentgenol.* **174**, 1657–1665 (2000). doi:10.2214/ajr.174.6.1741657 Medline

15. C. K. Zarins, D. P. Giddens, B. K. Bharadvaj, V. S. Sottiurai, R. F. Mabon, S. Glagov, Carotid bifurcation atherosclerosis. Quantitative correlation of plaque localization with flow velocity profiles and wall shear stress. *Circ. Res.* **53**, 502–514 (1983). doi:10.1161/01.RES.53.4.502 Medline

16. L. Boussel, V. Rayz, A. Martin, G. Acevedo-Bolton, M. T. Lawton, R. Higashida, W. S. Smith, W. L. Young, D. Saloner, Phase-contrast magnetic resonance imaging measurements in intracranial aneurysms in vivo of flow patterns, velocity fields, and wall shear stress: Comparison with computational fluid dynamics. *Magn. Reson. Med.* **61**, 409–417 (2009). doi:10.1002/mrm.21861 Medline

17. Y. Nakayama, *Introduction to Fluid Mechanics* (Butterworth-Heinemann, 2018).

18. maziarraissi. (2019, December 7). maziarraissi/HFM: Hidden Fluid Mechanics (Version v1.0). Zenodo. http://doi.org/10.5281/zenodo.3566161

19. Yazdani. (2019, December 9). alirezayazdani1/HFM: HFM - Synthetic Data Generators (Version v1.0). Zenodo. http://doi.org/10.5281/zenodo.3567215

20. M. Raissi, *J. Mach. Learn. Res.* **19**, 1 (2018).

21. A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, *J. Mach. Learn. Res.* **18**, 1 (2018).

22. M. Abadi *et al.*, *12th {USENIX} Symposium on Operating Systems Design and Implementation* ({OSDI} 16) (2016), pp. 265-283.

23. A. Paszke, *et al.*, Automatic differentiation in PyTorch. *NIPS 2017 Workshop Autodiff Submission* (2017).

24. J. Håstad, On the Correlation of Parity and Small-Depth Circuits. *SIAM J. Comput.* **43**, 1699–1708 (2014). doi:10.1137/120897432

25. P. Ramachandran, B. Zoph, Q. V. Le, arXiv:1710.05941 (2017).

26. M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, J. S. Dickstein, *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (JMLR. Org, 2017), pp. 2847- 2854.

27. T. Salimans, D. P. Kingma, *Adv. Neural Inf. Process. Syst.* **2016**, 901–909 (2016).

28. S. Ioffe, C. Szegedy, *International Conference on Machine Learning* (2015), pp. 448-456.

29. D. P. Kingma, J. Ba, *Proceedings of the 3rd International Conference on Learning Representations (ICLR)* (2015).

30. S. H. Rudy, S. L. Brunton, J. L. Proctor, J. N. Kutz, Data-driven discovery of partial differential equations. *Sci. Adv.* **3**, e1602614 (2017). doi:10.1126/sciadv.1602614 Medline

31. G. Cybenko, Approximation by superpositions of a sigmoidal function. *Math. Contr. Signals Syst.* **2**, 303–314 (1989). doi:10.1007/BF02551274

32. S. J. Pan, Q. Yang, A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* **22**, 1345–1359 (2009). doi:10.1109/TKDE.2009.191

33. M. Van Dyke, *An Album of Fluid Motion* (The Parabolic Press, 1982).

34. X. Ma, G. E. Karniadakis, A low-dimensional model for simulating three-dimensional cylinder flow. *J. Fluid Mech.* **458**, 181–190 (2002). doi:10.1017/S0022112002007991

35. G. E. Karniadakis, G. S. Triantafyllou, Three-dimensional dynamics and transition to turbulence in the wake of bluff objects. *J. Fluid Mech.* **238**, 1–30 (1992). doi:10.1017/S0022112092001617

36. G. K. Hansson, Inflammation, atherosclerosis, and coronary artery disease. *N. Engl. J. Med.* **352**, 1685–1695 (2005). doi:10.1056/NEJMra043430 Medline

37. L. Grinberg, G. E. Karniadakis, Outflow boundary conditions for arterial networks with multiple outlets. *Ann. Biomed. Eng.* **36**, 1496–1514 (2008). doi:10.1007/s10439-008-9527-7 Medline